АВТОМАТИКА, ТЕЛЕМЕХАНИКА И СВЯЗЬ

УДК 004.312.466

К. А. БОЧКОВ, доктор технических наук, С. Н. ХАРЛАП, кандидат технических наук, Б. В. СИВКО, магистр технических наук, Белорусский государственный университет транспорта, г. Гомель

ОСОБЕННОСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ МИКРОПРОЦЕССОРНЫХ СИСТЕМ ЖЕЛЕЗНОДОРОЖНОЙ АВТОМАТИКИ И ТЕЛЕМЕХАНИКИ

Рассмотрены особенности программного обеспечения микропроцессорных систем железнодорожной автоматики и телемеханики в контексте их влияния на проектирование и верификацию критически важных объектов информатизации. К рассматриваемым особенностям относятся: распределенные аппаратно-программные комплексы, режим эксплуатации 24/7, системы реального времени, безопасное состояние, стратегии обеспечения безопасности, независимость времени функциональной работы от технологической ситуации, многоканальная обработка, диверситет, конфигурирование, отладка и эксплуатация.

Приведены примеры использования особенностей для разработки и верификации безопасного программного обеспечения аппаратно-программных комплексов.

Поиск новых методов и средств для создания безопасного программного обеспечения (ПО) в инженерии необходим, но затруднителен из-за высокой сложности систем и их разнообразия. В отличие от производства разработчики ПО не создают один и тот же продукт, а каждый раз действуют в соответствии с заданными спецификациями, поэтому каждая новая реализация является отдельной разработкой и отличается от всех предыдущих. При этом процесс создания ПО полностью не формализуется, а качество продукта зависит от человеческого фактора и организации каждого из этапов жизненного цикла всего аппаратно- программного комплекса (АПК) [1–3].

Каждая микропроцессорная система создается в условиях зависимости от большого числа контекстуальных факторов, поэтому сделать обстоятельные и общие выводы на основании изучения некоторых из них затруднительно, и сложно предсказать, как поведет та или иная методика в случае, если ПО работает или разрабатывается в другом окружении. В связи с этим важно при разработке и верификации понимать особенности эксплуатации и использовать подходящие методы и средства. При этом применение более близких к рассматриваемой предметной области способов позволяет достичь лучших результатов [4]. Но, с другой стороны, возможен обратный процесс, когда на основании известных свойств предметной области ведется поиск и апробация различных подходов для решения поставленных задач.

Системы железнодорожной автоматики и телемеханики (СЖАТ) относятся к критичным по безопасности системам (*safety-critical systems*) и имеют свои особенности, отражающиеся на всем жизненном цикле ПО, которые могут быть принципиально отличными от свойств других систем, к которым предъявляются повышенные требования по безопасности и надежности функционирования (*mission-critical/business-critical systems*) [5, 6]. Кроме того, сами СЖАТ могут быть как различными, так и отличаться от связанных с безопасностью систем других отраслей промышленности.

Выделение отличительных особенностей проектируемых микропроцессорных АПК позволяет эффективно использовать подходящие методы и средства как в соответствии с предметной областью, так и в отношении заданных спецификаций конкретной задачи.

Каждая из особенностей представляет собой некоторое характерное условие, которому микроэлектронная система может удовлетворять или не удовлетворять. При проектировании или верификации критически важных систем информатизации можно просмотреть список нижеописанных особенностей и, в случае выполнения рассматриваемым АПК любого из условий списка, обратить особое внимание на сопутствующие выводы. Описанная последовательность действий позволит облегчить формализацию и дать более точное определение функции безопасности, а также предоставить готовые решения и рекомендации для этапов проектирования и разработки. Описанный подход положительно влияет на формирование полноты критериев опасного отказа технического задания и на верификацию уже разработанных критически важных систем информатизации.

Представленные особенности и их характерные отличия основаны на опыте проектирования, разработки и верификации микропроцессорных АПК, проводимых в лаборатории «Безопасность и ЭМС технических средств» Белорусского государственного университета транспорта. В данной статье рассматриваются только те особенности, которые влияют на проектирование и верификацию ПО.

Распределенные аппаратно-программные комплексы. Понятие распределенной системы зависит от того контекста, в котором оно используется [7]. СЖАТ являются системами, которые распределены по всей железной дороге, а также работают в различных условиях [8]. Однако с точки зрения разработки и верификации ПО железнодорожных АПК наиболее важным свойством является необходимость обеспечения безопасности эксплуатируемых систем, большинство из которых не могут быть монолитными и являются ком-

позицией множества подсистем, выполняющих более низкоуровневые функции.

Таким образом, при разработке сложных СЖАТ для каждой из них формируется глобальное условие безопасности (глобальный инвариант), которое должно выполняться для всего АПК. Например, эксплуатируемая система не должна допускать перевода стрелки под движущимся поездом. Однако данное условие обеспечивается не одним устройством, а всем комплексом в целом. При этом устройства связи с напольными объектами выполняют свое локальное условие безопасности (локальный инвариант) – безопасную передачу управляющего воздействия или безопасное снятие информации с напольного объекта. В то же время ядро микропроцессорной централизации обеспечивает такие функции, как безопасную установку маршрутов или безопасную передачу информации на устройства отображения, при этом выполняя уже свой локальный инвариант.

Данное построение распределенной безопасной системы позволяет эффективно проводить декомпозицию, использовать множество различных устройств на одной станции и использовать их повторно на другой. Кроме того, декомпозиция облегчает разработку и верификацию больших и сложных систем, позволяя при этом использовать дополнительные методы, такие как самостабилизация (*self-stabilization*) или контрактное программирование (*design by contract*) [9, 10].

Таким образом, при разработке и верификации больших и сложных СЖАТ эффективным методом является проведение декомпозиции рассматриваемой системы, при которой для каждой из используемых систем проводится верификация, а в дальнейшем на её основе возможна верификация всего комплекса в целом. В отсутствие декомпозиции выполнить такую операцию сложно.

Режим работы 24/7. Типичный режим работы СЖАТ – 24 часа в сутки, 7 дней в неделю. В связи с этим разрабатываемое ПО должно быть приспособлено к длительной эксплуатации [8]. В таком случае необходимо проектировать и верифицировать систему таким образом, чтобы её безопасность или функциональность не изменялась в течение длительных промежутков времени, а в идеале выполняла возложенные на неё задачи на протяжении всего жизненного цикла эксплуатации.

В критически важных объектах информатизации могут использоваться АПК другого типа, когда алгоритмы работают не постоянно, а обрабатывают конкретные входные данные. Например, в системах мониторинга напольных устройств СЖАТ при выполнении единичной диагностики [11].

Если рассматривается система, работающая длительное время, то для нее может проявиться накопление ошибок, зависимость от предыдущих состояний, рассинхронизация по времени и др. [12] В качестве примера можно рассмотреть спроектированную для 14-часовых дежурств систему *Patriot*, которая в случае непрерывной работы длительностью более 100 часов перестала выполнять возложенные на неё функции из-за накопительной ошибки, что привело к отказу системы и человеческим жертвам [13]. Другим примером является потеря космического аппарата *Deep Impact* по причине переполнения счетчика времени [14].

Для систем, обладающих качеством 24/7, необходимым является доказательство того, что АПК сохраняет свой внутренний инвариант по функциональности и безопасности поведения бесконечное время или время, которое намного больше возможного срока эксплуатации. Кроме того, рекомендуется выносить определение инварианта на этап проектирования или валидации.

Как показывает практика, алгоритмы ПО микроэлектронных устройств СЖАТ, которые выполняются только один раз, встречаются редко, и для большинства рассматриваемых систем необходимо обращать внимание на длительную их эксплуатацию.

Системы реального времени. Микропроцессорные железнодорожные АПК должны выдерживать определенные критерии временны*х* параметров функционирования: гарантированный переход в безопасное состояние по тайм-ауту, осуществление периодического обновления устройств индикации, обеспечение длительности реакции на внешние воздействия и задержек во времени программным способом и др. К каждому из критериев предъявляются соответствующие, зависящие от спецификации, требования. Для их выполнения могут привлекаться как аппаратные, так и программные решения, но каждое из них подлежит верификации и рекомендуется закладывать их выполнение ещё на этапах проектирования и валидации.

В настоящее время для систем реального времени используется два альтернативных подхода: циклическое выполнение (*cyclic executive*) и выполнение задач с заданными приоритетами (*fixed priority executives*) [15]. Каждый из подходов имеет свои преимущества и недостатки, а также могут быть использованы гибридные варианты.

У критичных по безопасности систем в случае использования циклического выполнения существуют следующие проблемы [15, 16]:

– необходимость гарантии завершения цикла в фиксированное время;

– сложности при реализации наслоения задач друг на друга и на разные циклы;

– необходимость разбиения больших задач в рамках нескольких периодов циклов;

– сложность добавления новых задач в уже созданную систему.

Практика показывает, что в СЖАТ время выполнения каждого из циклов позволяет реализовывать необходимый объем функциональности в заданное время [17]. Сами же системы проектируются по V-модели, когда добавление новых задач в существующие системы либо не проводится, либо происходит перепроектирование [3]. Данное обстоятельство связано с тем фактом, что внесение изменений в ПО без прохождения фаз проектирования ведет к увеличению числа скрытых дефектов [18, 19].

Циклическая система обладает достоинством, которое крайне важно для СЖАТ, связанных с безопасностью: хороший уровень детерминизма [16]. Данное свойство позволяет создавать АПК, в которых можно точно предсказать поведение системы и время выполнения цикла [20].

Системы реального времени, построенные с использованием подхода выполнения задач с приоритетами,

более сложны в верификации. При большом количестве задач возникают существенные сложности как в расстановке приоритетов, так и в доказательстве безопасности при любом сценарии поведения [16]. Кроме того, как правило, в таких системах создается отдельный слой, управляющий переключением контекстов выполняемых задач, – операционная система реального времени, которая усложняет систему в целом и требует отдельных шагов верификации и доказательства безопасности.

Мировая практика показывает, что системы задач с заданными приоритетами рекомендуется использовать в случае, если окружение изменяется быстро и непредсказуемо, а также требуется быстрое реагирование на изменяющиеся обстоятельства. Если же окружение предсказуемо и изменяется медленнее, чем реагирует на это АПК, то рекомендуется использовать циклические системы [16].

Безопасное состояние. В СЖАТ, как правило, имеется возможность перехода в безопасное состояние в случае обнаружения опасного отказа. В ряде критичных по безопасности систем такая возможность отсутствует или не декларируется, но в СЖАТ возможность перехода в безопасное состояние, как правило, декларируется на уровне стратегии обеспечения безопасности всей системы вне зависимости от элементной базы или реализуемой функциональности. Данный переход характерен тем, что в безопасном состоянии система теряет способность выполнять некоторые из своих функций.

В ряде критичных по безопасности систем описываемое условие не выполняется. Например, в случае отказа микропроцессорного кардиостимулятора у пациента с нарушениями сердечного ритма переход в пассивное состояние может привести к летальному исходу [21]. Аналогичное свойство может быть характерно для аппарата искусственного кровообращения при его работе во время операции на открытом сердце [21]. Полная или частичная потеря функциональности автоматизированной системы управления в больнице скорой помощи может стать причиной смертельных случаев [22, 23].

В системах, где предъявляются повышенные требования по безопасности и надежности функционирования, в случае необходимости выполнения определенной цели (*mission-critical systems*) во время отказов и сбоев основной задачей является не обеспечение безопасности с помощью перехода в особое состояние, а повышение вероятности выполнения задачи [24].

Для СЖАТ допустима либо частичная потеря функциональности (потеря контроля над частью станции или переход на ручное управление), либо полная на малые промежутки времени, но только для сохранения требуемого уровня безопасности. Такое поведение является одним из элементов общей стратегии безопасности, и оно учитывается при проектировании и верификации железнодорожных АПК. Например, при разработке устройств связи с напольными объектами подразумевается, что в случае опасного отказа каждое из них может выйти из строя, но при этом оно обязано перевести управляющие ключи в безопасное состояние. В дальнейшем в течение некоторого короткого времени проблемный блок должен быть найден и заменен.

Таким образом, большинство СЖАТ могут переходить в безопасное состояние, и соответственно, при их проектировании, реализации и верификации может быть использован описанный способ, который представляет собой единую фундаментальную базу, позволяющую применять общие методы построения модулей и подсистем различных железнодорожных АПК [25]. Таким образом, данный подход уменьшает сложность проектирования и верификации, что отражается на качестве эксплуатации разработанных систем.

Стратегии обеспечения безопасности. В настоящее время на железной дороге имеется большой опыт построения безопасных систем и, как следствие, отработанных стратегий обеспечения безопасности, использующие общие принципы построения системы в целом, что позволяет их применять на ранних этапах разработки и на всех уровнях построения больших АПК [25, 26].

Данные обстоятельства обусловлены тем, что СЖАТ функционально изменяются медленно, и при этом возможно использование типовых решений [8, 27]. Это позволяет с учетом накопленного опыта обеспечивать более высокий уровень безопасности.

Примерами стратегий обеспечения безопасности являются: использование элементной базы с несимметричными характеристиками отказов и самопроверяемых схем; резервирование; снижение интенсивностей потока отказов элементов и др. [25]. Их применение формирует концепцию обеспечения безопасности для рассматриваемого АПК, которая определяет основные принципы обеспечения безопасного функционирования СЖАТ. Концепция формируется на основании таких факторов, как свойства элементной базы, средства контроля, структуры и алгоритмы работы системы.

Таким образом, во время разработки и верификации системы необходимо рассмотреть её свойства в целом и определить те основополагающие принципы, на основании которых будет строиться последующий процесс. Данный подход позволяет использовать готовые решения и теоретический базис, что упрощает выполнение задачи и улучшает качество разработки и верификации АПК.

Независимость времени функциональной обработки от технологической ситуации. В ряде систем, которые используются в различных предметных областях, когда требуется управление технологическими процессами в реальном времени, может иметь место период времени, когда на АПК подается более высокая нагрузка. Она может требовать более мощного вычислительного ресурса (необходимость более быстрого центрального процессора), больших расходов памяти (для хранения обрабатываемых в настоящее время данных), высокой производительности базы данных и др. Например, в телекоммуникационных, платежных и пользовательских сервисах могут наблюдаться пики нагрузки в рабочее, обеденное или вечернее время, а ночью нагрузка может практически отсутствовать [28, 29]. Для *mission-critical-systems* такое свойство также может быть характерно; например, при посадке марсохода *Curiosity* потребовалось использование мощностей всех процессоров, а в последующем такие мощности уже не являлись необходимыми [24].

В СЖАТ окружающая обстановка с точки зрения ПО изменяется редко и медленно, поэтому есть возможность создавать системы таким образом, чтобы

нагрузка на её компоненты была неизменной. Кроме того, для железнодорожных систем в сущности нет задач, характерных для *mission-critical systems*, когда необходимо повысить вероятность выполнения задачи, а не обеспечить безопасность работы системы. В связи с этим железнодорожные АПК проектируются таким образом, чтобы все его компоненты имели такой запас прочности, когда подаваемая нагрузка не превышает заранее заданного на этапе проектирования предела. Данное свойство, заданное на этапе проектирования и верифицируемое позже, позволяет системе гарантированно выполнять свои задачи в реальном времени, обеспечивая тайм-ауты, периоды обработки и предельные времена отклика. Примером системы, у которой все связанные с безопасностью элементы обладают описанным свойством, является микропроцессорная централизация стрелок и сигналов «ИПУТЬ» (МПЦ «ИПУТЬ») [30].

Таким образом, во время рассмотрения СЖАТ важно выделение факта того, что отсутствует необходимость как можно более быстрого реагирования на технологическую ситуацию, а присутствует необходимость обеспечения гарантированной обработки в любых условиях за заданное предельное время. Если оно выполняется на этапе проектирования или на ранних этапах верификации, то это обеспечивает качественно лучшую гарантию независимости времени функциональной обработки от технологической ситуации.

Многоканальная обработка. Основным способом обеспечения высокой функциональной безопасности СЖАТ является многоканальная обработка, когда все ответственные операции выполняются параллельно во времени в нескольких каналах с последующим сравнением полученных результатов. Обработка считается успешной, если результаты, полученные в разных каналах, совпадают. При этом сравнение результатов может выполняться как аппаратно, так и программно.

При разработке и верификации программного обеспечения, предназначенного для последующей многоканальной обработки, можно выделить два ключевых момента, оказывающих значительное влияние на безопасность: синхронизация работы каналов и использование программного сравнения результатов.

Основная задача синхронизации работы каналов – обеспечить одновременное поступление контрольной информации из разных каналов на сравнение.

Даже при использовании идентичных аппаратных и программных средств в разных каналах, если не применять дополнительных мер по синхронизации работы каналов, то после нескольких месяцев непрерывной работы рассинхронизация может достигать десятков циклов работы системы. Это может привести к тому, что на схему сравнения поступят данные из разных тактов работающих каналов. В большинстве случаев такая ситуация приведет к несанкционированному срабатыванию схемы сравнения с последующим отключением системы, что не является опасным отказом. Однако не исключена ситуация и опасного проявления рассинхронизации, заключающаяся в замедлении реакции системы на критические воздействия. К проблемам синхронизации можно также отнести контроль актуальности предоставляемой каналами информации и реакцию системы на пропадание связи с параллельным каналом.

Программное сравнение позволяет сократить аппаратные затраты, применять сложные алгоритмы анализа контрольной информации, использовать кодирование, увеличить количество контролируемых параметров. Однако при этом резко повышаются требования к безопасности программного обеспечения и усложняется его последующая верификация. Необходимо исключить возможность формирования сообщения о равенстве данных из разных каналов при их фактическом неравенстве под действием различных возмущающих факторов [31]. Для исключения такого события на выходе модуля сравнения формируется динамический сигнал поддержания системы в работоспособном состоянии, который может быть достаточно сложным.

Таким образом, как на этапе разработки, так и при верификации программного обеспечения обязательным является подробный анализ программного кода, отвечающего за синхронизацию и сравнение, с учетом возможных искажений программного кода, контрольной информации и отказов аппаратуры.

Диверситет. Целью *n*-версионного или диверситетного программирования является обнаружение оставшихся ошибок проектирования программного обеспечения и отказов аппаратных средств с целью предотвращения опасных отказов.

N-версионное программирование предусматривает *n*-разовую реализацию одной и той же программы различными способами. Эффективность данного метода определяется, прежде всего, степенью различия (непохожести) программных компонент (диверситетом). Чем выше диверситет, тем ниже вероятность появления одинаковой реакции различных версий программ при нарушении работы технических средств или наличии программных ошибок.

Диверситет программ может быть достигнут (в порядке убывания степени различия):

– созданием различных версий программы разными программистами или коллективами программистов;

– использованием упрощенной модели программы в качестве другой версии;

– использованием разных методов логической организации программ (алгоритмов решения задач);

– применением различных языков программирования или разных версий компиляторов с одного и того же языка, что позволяет защититься от некоторых ошибок, связанных с организацией данных, работой с памятью и проверками, выполняемыми компиляторами по умолчанию. Однако данный способ не позволяет обнаружить алгоритмические ошибки.

Методы, базирующиеся на использовании различных алгоритмов решения задачи и упрощенной модели программы в качестве другой версии, требуют искусственного внесения различий на этапе проектирования ПО. Поэтому, хотя данные методы и позволяют обнаруживать алгоритмические ошибки, величина диверситета при их использовании во многом зависит от того, насколько правильно выбраны алгоритмы и модели.

Самым лучшим способом достижения диверситета, рекомендованным международными стандартами [25], является создание различных версий программы разными программистами или коллективами программистов. При этом предполагается, что при создании бо-

лее-менее сложного программного обеспечения независимыми программистами обязательно будут допущены ошибки, но эти ошибки будут по-разному проявляться при эксплуатации. Поэтому они гарантированно будут обнаружены контрольными средствами и не смогут привести к возникновению опасной ситуации.

Результаты работы различных версий программ обязательно сравниваются контрольными средствами. Сравнение может быть организовано аппаратно или программно. В последнем случае для корректного сравнения результатов работы контрольная программа должна быть безопасной.

Данный подход позволяет контролировать как состояние технических средств, так и наличие программных ошибок. К недостаткам *n*-версионного программирования можно отнести значительное увеличение программы и затрат на его разработку и верификацию. Одним из наиболее сложных вопросов при этом является оценка достаточности диверситета.

Конфигурирование, отладка и эксплуатация. В системах, критичных к безопасности, важна организация процесса разработки и сопровождения, и при этом требуется обеспечение высокого уровня безопасности [3]. В связи с этим присутствуют строгие требования по процессу отладки и конфигурирования, когда каждое изменение в системе означает перепроектирование, поэтому конфигурация в процессе эксплуатации не может изменяться часто, а процесс отладки является длительным. Данные условия обеспечиваются при высоком уровне повторного использования имеющихся модулей, так как большое количество ресурсов тратится на разработку и верификацию каждого из них.

В связи с этим при разработке микропроцессорных СЖАТ важно не только использование имеющегося теоретического базиса, но и создание систем, которые могут быть адаптированы к различному окружению. Например, при разработке микропроцессорной централизации можно выделить подсистемы, которые не зависят от конкретной станции, и использовать их повторно для новых станций. Примерами таких подсистем в отношении МПЦ «ІПУТЬ» могут являться [30]:

– ядро МПЦ;

– блоки управления и сигнализации;

– ПО, выполняющее функцию интерфейса для дежурного по станции;

– ПО, обеспечивающее связь между устройствами посредством различных телекоммуникационных интерфейсов.

Все описанные подсистемы могут быть спроектированы таким образом, что в зависимости от конфигурации их можно задействовать на любой станции, при этом общая функциональность используется несколько раз и не требует повторного проектирования и более глубокой верификации.

Следующим шагом для повторного применения уже разработанной системы является составление типовых шаблонов, позволяющих выполнять интеграцию подмножества систем в соответствующей ситуации.

Таким образом, при проектировании СЖАТ в большинстве случаев используется уже ранее опробованное решение, которое имеет возможность конфигурирования для функционирования в новых условиях. Впоследствии система в течение длительного времени проходит эксплуатационные испытания, и только по прохождению всех этапов верификации сдается в эксплуатацию, в течение которой изменения являются редкими и требуют повторного проектирования и отладки.

В заключение следует сказать, что рассмотренные особенности ПО СЖАТ рекомендуется учитывать на ранних этапах проектирования, что позволяет создавать контролепригодные АПК, верификация которых в дальнейшем будет значительно облегчена. Данное качество имеет большое значение для любых критически важных объектов информатизации, так как это способствует повышению их качества разработки, улучшению доказательства безопасности и последующей эксплуатации.

Также следует отметить, что рассмотренные особенности ПО СЖАТ требуют специальных методов разработки и анализа на безопасность их функционирования. Методы, используемые в смежных областях и рекомендованные международными стандартами, в частности IEC 61508, имеют ограничения на применение и для железнодорожных систем могут быть использованы только после их адаптации.

Список литературы

1 **Lutz, R. R.** Analyzing software requirements errors in safety-critical, embedded systems / R. R. Lutz // Requirements Engineering. – Proceedings of IEEE International Symposium. – 1993. – P. 126–133.

2 **Leveson, N**. Safeware: System Safety and Computers / N. Leveson // Boston, USA, Addison-Wesley. – 1995.

3 **Смит, Д. Дж.** Функциональная безопасность. Простое руководство по применению стандарта МЭК 61508 и связанных с ним стандартов / Д. Дж. Смит, К. Дж. Л. Симпсон. – М. : Издательский дом «Технологии». – 2004. – 208 с.

4 **Smith, D. J.** Developments in the Use of Failure Rate Data and Reliability Prediction Methods for Hardware / D. J. Smith // Aerospace Engineering, Dissertation. – 2000.

5 **Knight, J. C.** Safety critical systems: challenges and directions / J. C. Knight // ICSE '02 Proceedings of the 24th International Conference on Software Engineering. – 2002. – P. 547–550.

6 **Sommerville, I.** Software engineering / I. Sommerville // Addison-Wesley. – 2007. – P. 44.

7 **Lamport, L.** Solved problems, unsolved problems and non-problems in concurrency / L. Lamport // ACM SIGOPS Operating Systems. – 1985. – Vol. 19, Is. 4. – P. 34–44.

8 **Сапожников, В. В.** Методы построения безопасных микроэлектронных систем железнодорожной автоматики. / В. В. Сапожников, Х. А. Христов, Д. В. Гавзов; под ред. В. Сапожникова. – М. : Транспорт, 1995. – 272 с.

9 **Dolev, S.** Self-stabilization / S. Dolev // Cambridge, Massachusetts, USA, MIT Press. – 2000.

10 **Meyer, B.** Applying "Design by Contract" / Bertrand Meyer // Computer. – 1992. – Vol. 25, Is. 10. – P. 40–51.

11 Диагностирование устройств железнодорожной автоматики и агрегатов подвижных единиц : учебник / А. Б. Бойник [и др.]. – Х. : ЧП Изд. «Новое слово». – 2008. – 304 с.

12 **Pidgeon, N.** Man-made disasters: why technology and organizations (sometimes) fail / N. Pidgeon, M. O'Leary // Safety Science. – 2000. – Vol. 34, Is. 1–3. – P. 15–30.

13 **Blair, M.** Patriot Missile Defense: Software Problem Led to System Failure at Dhahran, Saudi Arabia / M. Blair, S. Obenski,

P. Bridickas // Washington, D.C. : U.S. General Accounting Office. – 1992.

14 **Vergano, D.** NASA Declares End to Deep Impact Comet Mission / D. Vergano // National Geographic. – September, 2013. – [Электронный ресурс]. – Режим доступа: http://news.nationalgeographic.com/news/2013/09/130920-deep-impact-ends-comet-mission-nasa-jpl/.

15 **Locke, C. D.** Software architecture for hard real-time applications: cyclic executives vs. fixed priority executives / C. D. Locke // Journal Real-Time Systems. – 1992. – Vol. 4, Is. 1. – P. 37–53.

16 **Rushby, J.** Critical System Properties: Survey and Taxonomy / J. Rushby // Reliability Engineering and System Safety. – 1994. – Vol. 43, No. 2. – P. 189–219.

17 **Бочков, К. А.** Оценка временных параметров функционирования микропроцессорных устройств связи с объектами систем железнодорожной автоматики и телемеханики / К. А. Бочков, С. Н. Харлап, Б. В. Сивко // Вестник БелГУТа: Наука и транспорт. – 2012. – № 2 (25). – С. 12–15.

18 **Khoshgoftaar, T. M.** The Impact of Software Evolution and Reuse on Software / T. M. Khoshgoftaar, E. B. Allen, K. S. Kalaichelvan, N. Goel // Empirical Software Engineering. – 1996. – Vol. 1, Is. 1 – P. 31–44.

19 **Adams, E. N.** Optimizing preventive service of software products / E. N. Adams // IBM Journal of Research and Development archive. – New York. – 1984. – Vol. 28, Is. 1. – P. 2–14.

20 **Харлап, С. Н.** Верификация циклических систем железнодорожной автоматики и телемеханики / С. Н. Харлап, Б. В. Сивко // Вестник БелГУТа: Наука и транспорт. – 2013. – № 2 (27). – С. 37–41.

21 **Peterson, I.** Fatal Defect: Chasing Killer Computer Bugs / I. Peterson // New York, USA. Times Books. – 1995.

22 **Arthur**, C. Ambulance computer system was 'to complicated' / C. Arthur // New Scientist. – 1992.

23 **Blyth**, **A.** Issues arising from medical system's failure / A. Blyth // ACM SIGSOFT Software Engineering Notes. – 1997. – Vol. 22, Is. 2. – P. 85–86.

24 **Holzmann**, **G. J.** Mars Code / G. J. Holzmann // Communications of the ACM. – 2014. – Vol. 57, No. 2. – P. 64–73.

25 **Бочков, К. А.** Микропроцессорные системы автоматики на железнодорожном транспорте : учеб. пособие / К. А. Бочков, А. Н. Коврига, С. Н. Харлап. – Гомель : БелГУТ, 2013.

26 **Сапожников, В. В.** Теория дискретных устройств железнодорожной автоматики, телемеханики и связи / В. В. Сапожников, Ю. А. Кравцов, Вл. В. Сапожников; под ред. В. В. Сапожникова, 2-е изд., перераб. и доп. – М. : УМК МПС России. – 2001. – 312 с.

27 **Гапеев, В. И.** Безопасность движения на железнодорожном транспорте / В. И. Гапеев, Ф. П. Пищик, В. И. Егоренко. – Мн. : Полымя, 1996. – 360 с.

28 Разработка высоконагруженных систем. По материалам конференции HighLoad 2010–2011 / О. Бунин [и др.]. – М. : Изд. Олега Бунина. – 2012.

29 **Bueno, C.** Mature Optimization Handbook / C. Bueno // Facebook. – 2013. – 89 p. – [Электронный ресурс]. – Режим доступа: http://carlos.bueno.org/optimization/.

30 Микропроцессорная централизация стрелок и сигналов МПЦ «ИПУТЬ» / К. А. Бочков [и др.] // Автоматика, связь, информатика. – 2009. – № 7. – С. 14–19.

31 **Липаев, В. В.** Функциональная безопасность программных средств / В. В. Липаев. – М. : СИНТЕГ, 2004. – 348 с.

Получено 08.05.2014

K. A. Bochkov, S. N. Kharlap, B. V. Sivko. Software Features of Microprocessor Railway Systems.

Software features of microprocessor railway systems have been considered in a context of their affect to a development and verification of safety-critical systems. A list of considered features: distributed systems of hardware and software, 24/7 service, real-time systems, safe state, safety strategies, processing time work independence from any situation, redundancy systems, diversity, configuration, debugging and maintenance.

Examples of using the features for the safety software development and verification have been included.